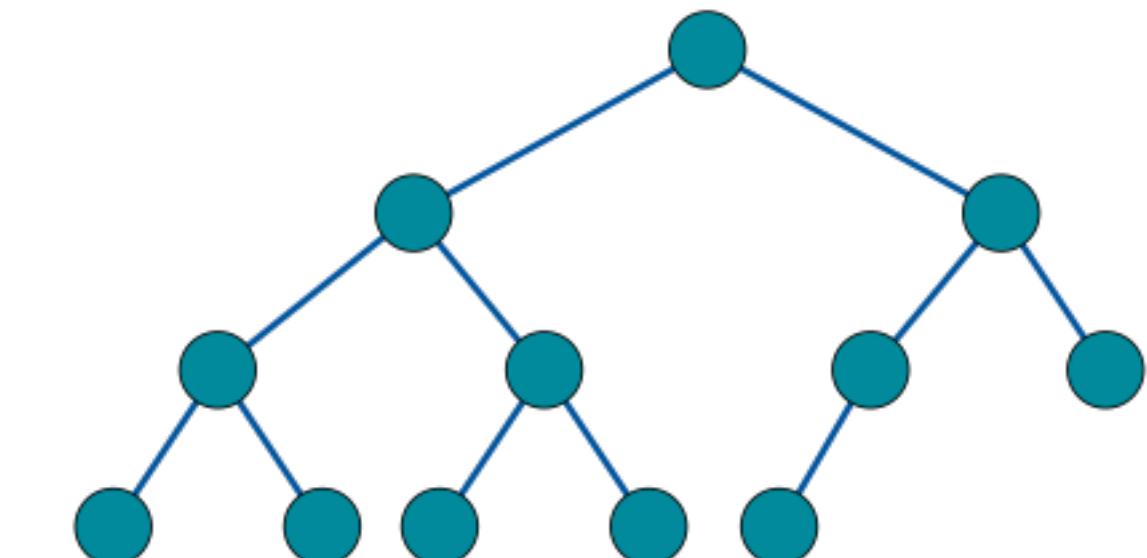
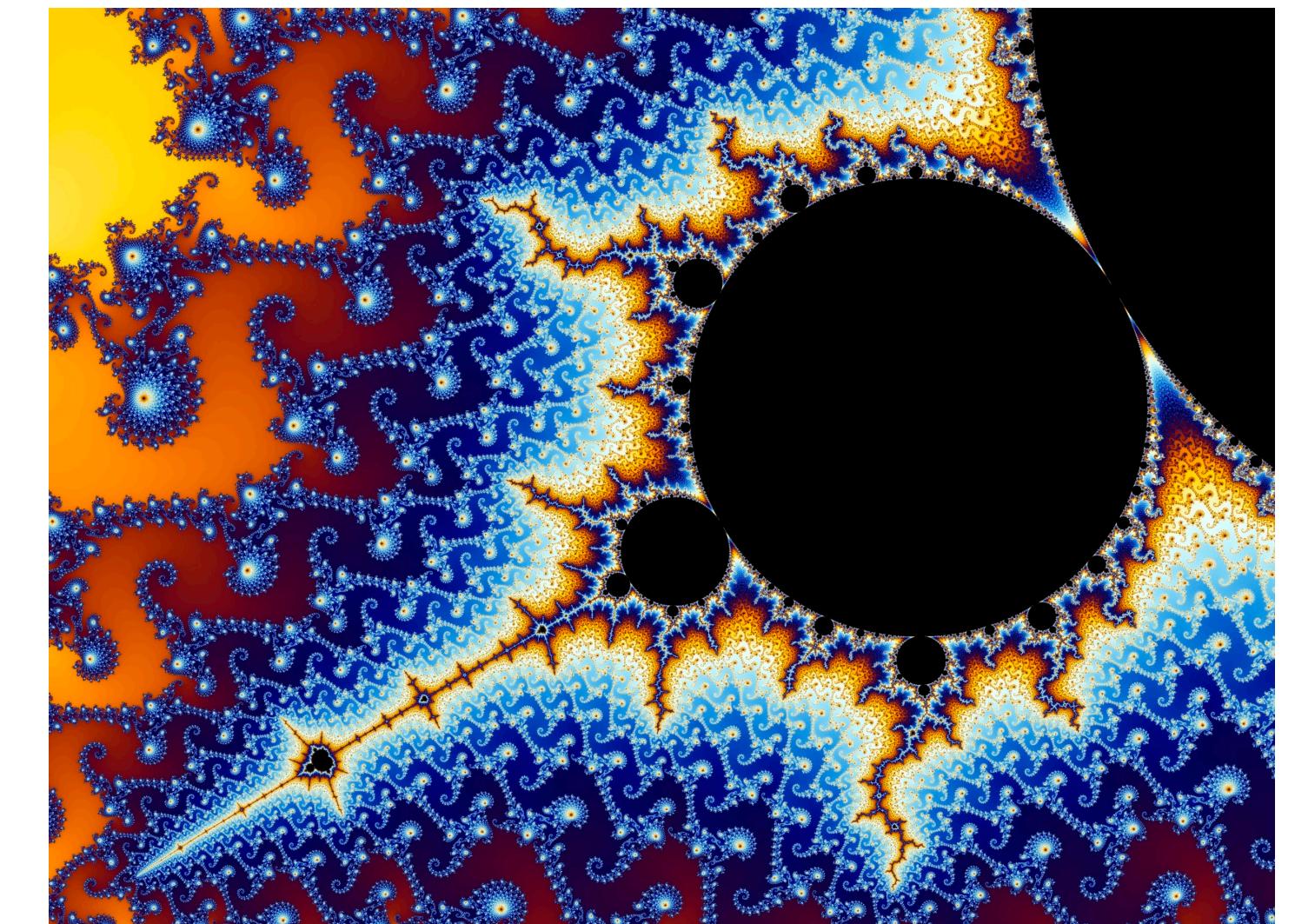


# Рекурсия

Михаил Каменщиков



# План занятия

- Познакомимся с рекурсией на примере чисел Фибоначчи и научимся решать рекуррентные соотношения
- Рассмотрим применение рекурсии на реальной рабочей задаче
- Узнаем, что такое tail recursion и зачем она нужна
- Проверим континуум гипотезу построим множество всех подмножеств

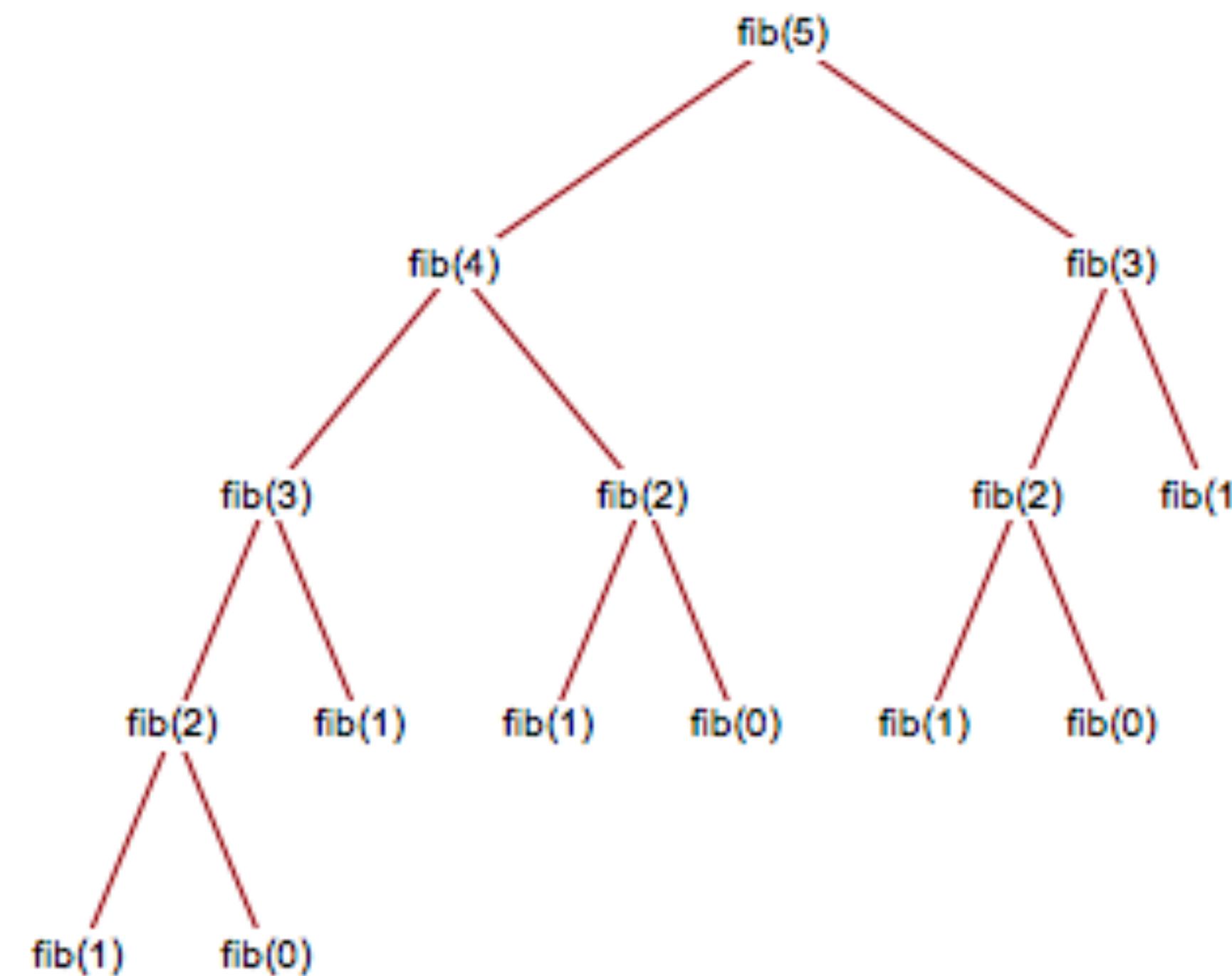
# Рекурсия

- Что такое рекурсия?
- Были ли у вас в жизни моменты, когда вам понадобилась рекурсия?

# Числа Фибоначчи

- $F_n = F_{n-1} + F_{n-2}$
- 1, 1, 2, 3, 5, 8, 13, 21 ...

# Дерево рекурсии



# Master theorem

- «Рецепт», который позволяет асимптотически оценить рекуррентные выражения вида  $T(n) = aT(n/b) + f(n)$

**Перерыв**

# Структура словаря

- Проблема - есть данные со сложной структурой, надо понять, как с ними работать

```
{  
    "result": {  
        "list": [  
            {  
                "type": "section",  
                "value": {  
                    "title": "Рекомендации",  
                    "engine": "personal_infinite",  
                    "engineVersion": "1",  
                    "list": [  
                        {  
                            "type": "item",  
                            "value": {  
                                "categoryId": 34,  
                                "service_item": {  
                                    "imagesCount": 10,  
                                    "locationId": 637640,  
                                    "microcatId": 302,  
                                    "price": 230000,  
                                    "sortTime": 1626781405,  
                                    "startTime": 1626781381,  
                                    "title": "Canyon grail cf sl 8.0",  
                                    "userId": 14723083,  
                                    "rawParams": null,  
                                    "isMarketplace": false,  
                                    "vector": [  
                                        74,  
                                        2,
```

# Tail Recursion

- Такая рекурсивная функция, последняя инструкция в которой - вызов функции.
- Преимущество - не нужно обратно возвращаться по стеку вызовов
- Компилятор может переделать это в итерацию (но не в питоне)



# Множество всех подмножеств

- Дано множество целых чисел, нужно вывести все его подмножества
- Пример:  $\{1, 2, 3\} \rightarrow \{1\}, \{2\}, \{3\}, \{1, 2\}, \{2, 3\}, \{1, 3\}, \{1, 2, 3\}, \{\}$

# Вопросы

- Где может быть полезна рекурсия?
- Как можно оценить сложность рекурсивной функции?
- Что делает функция на картинке?

```
def recursive(x: int, y: int):  
    if y == 0:  
        return 0  
    else:  
        return x + recursive(x, y - 1)
```